(0) Download the Binary (search) Tree StarterKit from moodle.

(1) Study the `Node` and `bTree` classes. We will do a ***walkthrough*** of the code I have given you.

(2) If you **insert** the following text strings into an empty binary tree in this order:

**`b, a, m, t, n, z, h, d, c`**

(a) **Draw the resulting tree**.

(b) Show the **output** of this tree if you were to print the nodes on one line in:

(i) **Pre-order** fashion:

(ii) **In-order** fashion:

(iii) **Post-order** fashion:

(c) What is the **height** of this tree where "height" will be defined here as the number of Nodes traversed from the root to the lowest leaf.

(d) **How many leaf nodes** are on this tree?

**Call me over to see your answers.**

---

(3) Study the **`bTree::InsertItem()`**`method.` See the code in `main()` that reads in words from the user at the keyboard (stdin) and inserts those words into the binary tree (you must really understand the `bTree::InsertItem()` method in order to write other methods for the `bTree` class.

(4) Write the **`bTree::printInorder()`**`method` to print the tree in an "in-order" fashion.

**Call me over to see your tree and in-order output. Is it correct?**

---

(5) Implement `printPreorder()` and `printPostorder()` methods for the `bTree` class.

**Call me over to see your new output. Is it correct?**

---

(6) Write the **`bTree::treeHeight()`** method.

(7) Write the **`bTree::leafCount()`** method.

(8) Write the **`bTree::search4Item()`** method.

(9) Write the **`bTree::~bTree()`** destructor (DTOR) method.